APPARATUS AND METHOD FOR GENERATING FINITE STATE TRANSDUCER FOR USE IN INCREMENTAL PARSING

1. Field of the Invention

[0001] The invention relates to an apparatus and a method for generating a finite state transducer for use in incremental parsing in real-time spoken language processing systems, a computer-readable recording medium storing a finite state transducer generating program, and an incremental parsing apparatus.

2. Description of the Related Art

[0002] Real-time spoken language processing systems such as a simultaneous interpretation system needs to recognize speech and make a response to the speech simultaneously. To achieve these processes, implementing parsing in order every time a fragment of speech is inputted, rather than implementing parsing after a whole sentence is inputted, is essential. This is referred to as incremental parsing.

[0003] As a framework for understanding sentence structures incrementally, several incremental parsing methods have been proposed so far. In incremental parsing, parse trees are generated from fragments of what have been inputted, even in the middle of speech. Thus, it is possible to understand a parse structure as of time of parsing at a stage where the input of the whole sentence is not completed. As the incremental parsing methods, Matsubara, et al., have proposed an incremental chart parsing algorithm in S. Matsubara, et al. "Chart-based Parsing and Transfer in Incremental Spoken Language Translation", Proceedings of NLPRS'97, pp.521-524 (1997). In this algorithm, context-free grammar rules are continuously applied to each input word, parse trees corresponding to each input word are generated, and connected with matching parse trees corresponding to each fragment of a sentence. However, the incremental chart-parsing algorithm has a problem that it is difficult to obtain sufficient performance on the real time performance required in the real-time spoken language processing systems.

[0004] To overcome the above problem in the incremental chart parsing algorithm, the inventors of the present invention have proposed an incremental parsing algorithm which uses finite state transducer in Minato et al., "Incremental Parsing using Finite State Transducer", Record of 2001 Tokai-Section Joint Conference of the Eighth Institute of Electrical and Related Engineers, Japan, P.279 (2001). This parsing algorithm can realize high speed parsing, since it executes parsing using a finite state transducer generated by approximate transformation of context-free grammars.

[0005] However, with the above parsing, as a result of approximate transformation, there is a problem that a sentence, which could be parsed with the original context-free grammar, cannot be parsed with the finite state transducer. The finite state transducer for use in the incremental parsing is generated by recursively replacing arcs in each network that represents grammar rules. However, owing to the limitation of memory size of a computer used to generate and/or to implement the finite state transducer, there are some cases where all arcs required for parsing cannot be replaced. As a result, the problem that a sentence, which could be parsed with the original context-free grammar, cannot be parsed with the finite state transducer occurs.

SUMMARY OF THE INVENTION

[0006] The present invention provides an apparatus and a method for generating a finite state transducer for use in incremental parsing capable of incrementally parsing a great number of sentences, a computer-readable recording medium storing a finite state transducer generating program, and an apparatus for incremental parsing.

[0007] According to one aspect of the invention, an apparatus for generating a finite state transducer for use in incremental parsing may include a recursive transition network creating device that creates a recursive transition network, the recursive transition network being a set of networks, each network representing a set of grammar rules based on a context-free grammar by states and arcs connecting the states, each arc having an input label and an output label, each network having a recursive structure where each transition labeled with a non-terminal symbol included in each of the networks is defined by another network; an arc replacement device that replaces an arc having an input label representing a start symbol included in the finite state transducer in an initial state by a network corresponding to the input label of the arc in the recursive transition network and further recursively repeats an arc replacement operation for replacing each arc, which is newly created from a replaced network, by another network in the recursive transition network; and a priority calculating device that calculates a derivation probability to derive a node of a parse tree corresponding to each of arcs whose input labels are non-terminal symbols in the finite state transducer based on statistical information regarding frequency of applying grammar rules and determines an arc replacement priority in terms of an obtained derivation probability. The arc replacement device continues applying the arc replacement operation to each arc included in the finite state transducer in descending order of the arc replacement priority until the finite state transducer reaches a predetermined size.

[0008] In the apparatus, the arc replacement operation is applied to the arcs in descending order of the arc replacement priority obtained based on the statistical information regarding the frequency of applying the grammar rules, thus reliably generating a finite state transducer capable of parsing a great number of sentences within the limited size.

[0009] The apparatus further includes an arc eliminating device that, after the application of the arc replacement operation by the arc replacement device terminates, eliminates arcs whose input labels are non-terminal symbols and further performs the arc replacement operation.

[0010] Therefore, in the apparatus, the arcs whose input labels are non-terminal symbols, which are not used for parsing, are eliminated and the arc replacement operation is concurrently performed, thus generating a finite state transducer capable of parsing a further great number of sentences.

[0011] In the apparatus, the derivation probability for a certain node represents a probability that grammar rules are applied in order to each node on a path from a root node to the certain node in the parse tree. The derivation probability P $(Xr_{M(1M)})$ for node $Xr_{M(1M)}$ may be determined as follows:

$$P(X_{r_{M(l_M)}}) = \prod_{i=1}^{M} \hat{P}(r_i \mid r_{i-N+1(l_{i-N+1})}, \ldots, r_{i-1(l_{i-1})})$$

wherein r_i represents a grammar rule, $r_i(l_i)$ represents that grammar rule r_i is applied and grammar rule r_i+1 to be applied next is applied to a node generated by the (l_i)-th element of the right side of r_i, and N is a predetermined positive integer.

[0012] The arc replacement operation is performed using the probability as an arc replacement order, thus reliably generating a finite state transducer capable of parsing a further great number of sentences.

[0013] According to another aspect of the invention, a computer-readable recording medium stores a program for generating a finite state transducer for use in incremental parsing. The program includes a recursive transition network creating routine that creates a recursive transition network, the recursive transition network being a set of networks, each network representing a set of grammar rules based on a context-free grammar by states and arcs connecting the states, each arc having an input label and an output label, each network having a recursive structure where each transition labeled with a non-terminal symbol included in each of the networks is defined by another network; an arc replacement routine that replaces an arc having an input label representing a start symbol included in the finite state transducer in an initial state by a network corresponding to the input label of the arc in the recursive transition network and further recursively repeats an arc replacement operation for replacing each arc, which is newly created from a replaced network, by another network in the recursive transition network; and a priority calculating routine that calculates a derivation probability to derive a node of a parse tree corresponding to each of arcs whose input labels are non-terminal symbols in the finite state transducer based on statistical information regarding frequency of applying grammar rules and determines an arc replacement priority in terms of an obtained derivation probability. In the program, the arc replacement routine continues applying the arc replacement operation to each arc included in the finite state transducer in descending order of the arc replacement priority until the finite state transducer reaches a predetermined size.

[0014] By causing the computer to execute the program, the arc replacement operation is applied to the arcs in descending order of the arc replacement priority obtained based on the statistical information regarding the frequency of applying the grammar rules, thus reliably generating a finite state transducer capable of parsing a great number of sentences within the limited size.

[0015] According to a further aspect of the invention, a method for generating a finite state transducer for use in incremental parsing may includes the steps of creating a recursive transition network, the recursive transition network being a set of networks, each network representing a set of grammar rules based on a context-free grammar by states and arcs connecting the states, each arc having an input label and an output label, each network having a recursive structure where each transition labeled with a non-terminal symbol included in each of the networks is defined by another network; replacing an arc having an input label representing a start symbol included in the finite state transducer in an initial state by a network corresponding to the input label of the arc in the recursive transition network and further recursively repeating an arc replacement operation for replacing each arc, which is newly created from a replaced network, by another network in the recursive transition network; and calculating a derivation probability to derive a node of a parse tree corresponding to each of arcs whose input labels are non-terminal symbols in the finite state transducer based on statistical information regarding frequency of applying grammar rules and determines an arc replacement priority in terms of an obtained derivation probability. In the step of replacing an arc, the arc replacement operation is continued applying to each arc included in the finite state transducer in descending order of the arc replacement priority until the finite state transducer reaches a predetermined size.

[0016] With the method, the arc replacement operation is applied to the arcs in descending order of the arc replacement priority obtained based on the statistical information regarding the frequency of applying the grammar rules, thus reliably generating a finite state transducer capable of parsing a great number of sentences within the limited size.

[0017] According to another aspect of the invention, an incremental parsing apparatus that perform incremental parsing may include a finite state transducer generated by the method, that is, by applying the arc replacement operation to the arcs in descending order of the arc replacement priority obtained based on the statistical information regarding the frequency of applying the grammar rules, the finite state transducer outputting at least one piece of a parse tree as a result of a state transition when each word is inputted thereto; and a connecting device that sequentially connects each piece of the parse tree outputted by the finite state transducer.

[0018] Using the finite state transducer of a limited size approximately transformed from the context-free grammar, the incremental parsing apparatus can parse a great number of sentences.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] An embodiment of the invention will be described in detail with reference to the following figures wherein:

[0020] Fig. 1 is a block diagram showing an entire configuration of a finite state transducer generator according to an embodiment of the invention;

[0021] Fig. 2 shows an example of P_X representing a set of grammar rules;

[0022] Fig. 3 shows an example of M_X in a recursive transition network;

[0023] Fig. 4 shows that states in the recursive transition network are integrated;

[0024] Fig. 5 illustrates an initial finite state transducer M0, which is given first;

[0025] Fig. 6 shows an example of an arc replacement operation and an arc-to-node relationship;

[0026] Fig. 7 illustrates a process of applying grammar rules to derive a certain node;

[0027] Fig. 8 illustrates a set of grammar rules obtained from a parse tree;

[0028] Fig. 9 shows four examples explaining how arcs are continuously eliminated;

[0029] Fig. 10 is a block diagram showing an entire configuration of an incremental parsing apparatus according to an embodiment of the invention;

[0030] Fig. 11 shows an example of a parsing process for a Japanese sentence;

[0031] Fig. 12 shows examples of a parse tree represented by output symbols strings; and

[0032] Fig. 13 shows an example of a parsing process for an English sentence;

[0033] Fig. 14 shows examples of a parse tree represented by output symbols strings; and

[0034] Fig. 15 is a graph showing an experimental result (accuracy rate) of a parsing process.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0035] An embodiment of the invention will be described in detail with reference to the accompanying drawings.

[0036] The entire configuration of a finite state transducer generator 1 will be described in detail with reference to Fig. 1. The finite state transducer generator 1 is made up of a recursive transition network creating part 2, an arc replacement part 3, a priority calculating part 4, and an arc eliminating part 5. The finite state transducer generator 1 is connected to a statistical information storage device 11. If an arc replacement operation (described later) is not performed, the arc eliminating part 5 may be eliminated from the configuration of the finite state transducer generator 1.

[0037] The finite state transducer generator 1 is realized by a computer, which includes, for example, a central processing unit (CPU), read-only memory (ROM), random-access memory (RAM), a hard disk drive, and a CD-ROM unit. The finite state transducer generator 1 is structured wherein, for example, finite state transducer generating program designed to cause the computer to function as the recursive transition network generating part 2, the arc replacement part 3, the priority calculating part 4 and the arc eliminating part 5 are stored in the hard disk drive, and the CPU reads the finite state transducer program from the hard disk drive and executes the program. If statistical information as to frequency of applying grammar rules stored in a recording medium such as a CD-ROM is read by the computer in advance and stored in the hard disk drive, the hard disk drive functions as the statistical information memory storage 11. As the statistical information regarding the frequency of applying the grammar rules, advanced telecommunications research (ATR) speech database with parse trees (Japanese dialogue) can be used.

[0038] Next, contents of processes executed in each of the above parts making up of the finite state transducer generator 1 will be described with reference to the drawings.

[0039] Previous to the contents of processes performed in the finite state transducer generator 1, a finite automaton, a finite state transducer, and a context-free grammar will be defined. First, a finite automaton will be defined. A finite automaton is defined in the form of a 5-tuple (Σ, Q, q_0, F, E), where Σ is an alphabet finite set, Q is a finite set of states, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is the set of final states, and E is a finite set of arcs. In addition, E may be defined by: $E \subseteq Q \times \Sigma \times Q$.

[0040] Each finite automaton has one initial state and one or more final states and is a network where state transitions are made according to arc labels. When an arc is defined by (p, A, q) $\in$ E(p, $q \in Q$, $A \in \Sigma$), state p is referred to as a start point of the arc and state q is referred to as an end point of the arc.

[0041] Next, a finite state transducer will be defined. A finite state transducer is defined in the form of a 6-tuple (Σ_I, Σ_O, Q, q_0, F, E), wherein Σ_I and Σ_O are a finite set of input alphabets and a finite set of output alphabets respectively, Q is a finite set of states, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is the set of final states, and E is a finite set of arcs. In addition, E may be defined by: $E \subseteq Q \times \Sigma_I \times \Sigma_O \times E$

[0042] In a finite automaton, an input label is assigned to each arc. In a finite state transducer, an input label and an output label are assigned to each arc. In other words, each arc has an input label and an output label. In a finite state transducer, when an element of Σ_I is inputted, an element of Σ_O is outputted and a state transition is made. A system using a finite state transducer can both accept inputted symbol strings and output symbol strings corresponding to the inputted ones.

[0043] Finally, a context-free grammar will be defined. A context-free grammar is defined in the form of a 4-tuple (N, T, P, S_0), wherein N and T are a non-terminal symbol and a terminal symbol respectively, $S_0 \in N$ is a start symbol and a root node of a parse tree generated from the grammar, P is a set of grammar rules. Each rule is indicated by $A \rightarrow \alpha$ ($A \in N$, $\alpha = (N \cup T)^+$), which indicates A is replaced by α. Most natural language structures can be described by context-free grammars.

[0044] Processes of each part making up the finite state transducer generator 1 will be described. In the embodiment, a context-free grammar is represented by a recursive transition network. Each arc in the obtained recursive transition is replaced by another network, thereby obtaining the finite state transducer. The following are descriptions of processes performed in each part. First, a process of creating a recursive transition network in the recursive network creating part 2 will be described, followed by processes of generating the finite state transducer using a replacement operation in the recursive transition network performed in the arc replacement part 3, the priority calculating part 4, and the arc eliminating part 5.

[0045] (Process of creating a recursive transition network in the recursive network generating part 2)

[0046] A recursive transition network is a set of networks that allow transitions labeled with non-terminal symbols. The recursive transition network has a recursive structure where a transition labeled with a non-terminal symbol included in each of the networks is defined by another network. The recursive transition network and the context-free grammar have an equivalent analysis capability. The following is a description of a method to create a recursive transition network, which is equivalent to a context-free grammar, from the context-free grammar. In the created recursive transition network, each network represents a set of grammar rules based on a context-free grammar by states and arcs connecting the states.

[0047] When each grammar rule has category X in the left hand side, a network M_X representing a set of grammar rules P_X, is defined in the form of a 5-tuple (Σ, Q_X, i_X, F_X, E_X), wherein $\Sigma = T \cup N$, i_X is an initial state, F_X is a set of final states, $F_X = \{f_X\}$, Q_X is a finite set of states, and E_X is a finite set of arcs.

[0048] To represent an element of Q_X, a grammar rule with a dot symbol (·) is introduced. In the grammar rule with a dot symbol, a dot symbol is inserted into an arbitrary place of the right hand side of a grammar rule such as $X \rightarrow \alpha \cdot \beta$. For notation simplification, the grammar with a dot symbol is represented with a 3-tuple, the left hand side in the grammar rule, the left side of the dot symbol of the right hand side in the grammar rule, and the right side of the dot symbol of the right hand side in the grammar rule. For example, $X \rightarrow \alpha \cdot \beta$ is represented as (X, α, β). With the use of this representation, Q_X, which is a finite set of states, is defined by:

$$\begin{aligned} Q_X &= \{(X,\alpha,\beta) \mid X \rightarrow \alpha\beta \in P_X, \alpha, \beta \in (N \cup T)^{+}\} \\ &\cup \{i_X, f_X\} \end{aligned}$$

E_X, which is a finite set of arcs, is defined by:

$$\begin{aligned} E_X &= \{((X,\alpha,A\beta),A,(X,\alpha A,\beta)) \\ &\qquad \mid X \rightarrow \alpha A\beta \in P_X\} \\ &\cup \{(i_X,A,(X,A,\beta)) \mid X \rightarrow A\beta \in P_X\} \\ &\cup \{((X,\alpha,A),A,f_X) \mid X \rightarrow \alpha A \in P_X\} \\ &\cup \{(i_X,A,f_X) \mid X \rightarrow A \in P_X\} \end{aligned}$$

wherein $X \in N$, $A \in N \cup T$, α, $\beta \in (N \cup T)^+$.

[0049] For example, when P_X is a set of grammar rules shown in Fig. 2, M_X is a network shown in Fig. 3. A path from the initial state i_X to the final state f_X of the network M_X corresponds to a grammar rule in P_X. Therefore, when a symbol string on the right hand side of a grammar rule is inputted to a network M_X, a state transition from ix to f_X is made along a path in M_X corresponding to the grammar rule. In the embodiment, a recursive transition network M is defined as a set of networks M_X by:

$$\mathcal{M} = \{M_X \mid X \in N\}$$

[0050] (Process of simplifying the recursive transition network in the recursive transition network creating part 2)

[0051] A recursive transition network created above may include some arcs having equivalent start points and the same labels, which produce redundancy, and state transitions cannot be decisively made. Therefore, states are integrated based on a finite automaton minimization procedure. In other words, as to each network M_X ($X \in N$) in the recursive transition network, if states are convertible equivalently, they are integrated into one state. However, state integration is not allowed when the number of elements of Fx is two or more. This is to simplify the replacement operation of M_X.

[0052] Simplification of M_X is realized by integrating states according to steps shown in Table 1. First, step 1 is repeated until there is no change in M_X, so that states are integrated. Then, step 2 is repeated until there is no change in M_X. Symbols used in the following are q, q', $q'' \in Q_X$, $A \in \Sigma_I$.

Table 1: SIMPLIFICATION OF NETWORK Mx

Step 1	Integrate q' and q'' if there is an existence of q, $(q, A, q') \in E_X$, $(q, A, q'') \in E_X$, and $q', q'' \notin F_X$.
Step 2	Integrate q' and q'' if there is an existence of q that satisfies $((q', A, q) \in Ex$ and $(q'', A, q) \in E_X)$ or $((q', A, q) \notin F_X$ and $(q'', A, q) \notin F_X)$, wherein q' and q'' are states and $A \in \Sigma_I$.

[0053] Fig. 4 shows an example of the above described integration process. In step 1, states that are reached from the same state with a transition labeled with A are integrated. In step 2, two states that share the same transition destination state with a transition labeled with D and do not have any destinations labeled with other symbols are integrated. In the simplified recursive transition network, a state where a transition is made from a certain state with the same label includes one final state and one state different from the final state at most.

[0054] (Process of generating a finite state transducer using the recursive transition network in the arc replacement part 3)

[0055] A process of generating a finite state transducer using the recursive transition network created in the above-described process will be described. First, an initial finite state transducer M_0 may be defined by:

$$M_0 = (Q_0, \Sigma_I, \Sigma_O, i, F, E_0)$$

wherein $Q_0 = \{ i , f\}$, $\Sigma_I = N \cup T$, $\Sigma_0 \subset (([_N] * (\Sigma_I) * (_N]) *)$, $F=\{f\}$, $E_0=\{(i, S_0, S_0, f)\}$.

[0056] Fig. 5 shows the initial finite state transducer M_0. An arc in the initial finite state transducer M_0 is replaced by network M_{s0}, such that a new arc is created. The newly created arc is replaced by another network. This replacement operation is recursively repeated, so that a finite state transducer is obtained. The replacement operation is carried out for an arc whose input label is a non-terminal symbol. An arc having input label X is replaced by M_X.

[0057] A change in the finite state transducer before and after the replacement operation will be described. The finite state transducer obtained by repeating the replacement operation several times as to the finite state transducer M_0 is referred to as M_j. M_j is defined by $(Q_j, \Sigma_I, \Sigma_O, i, F, E_j)$. An arc, which is defined by $e= (qs, X, {}_{01}X_{0r}, q_e) \in E_j$ wherein $_{01}$, $_{0r}$ are a category series with a left bracket "([$_N$])*" and a category series with a right bracket "($_N$))*" in their output alphabets respectively, is replaced by M_x, thereby obtaining the finite state transducer M_j. M_j is generated by adding new states and arcs to Q_j and E_j. Therefore, as a set of states and a set of arcs change, M_j is defined as $(Q'_j, \Sigma_1, \Sigma_0, i, F, E'_j)$. Q'_j and E'_j can be defined by:

$$\begin{aligned} Q'_j &= Q_j \\ &\cup \{eq \mid q \in (Q_X - \{i_X, f_X\})\} \end{aligned}$$

$$\begin{aligned}E'_j &= (E_j - \{e\}) \\ &\cup \{(eq_1, A, A, eq_2) \mid (q_1, A, q_2) \in E_X\} \\ &\cup \{(q_s, A, o_l[_X A, eq_2) \mid (i_X, A, q_2) \in E_X\} \\ &\cup \{(eq_1, A, A_X]o_r, q_e) \mid (q_1, A, f_X) \in E_X\} \\ &\cup \{(q_s, A, o_l[_X A_X]o_r, q_e) \mid (i_X, A, f_X) \in E_X\}\end{aligned}$$

wherein $q_1 \neq i_X$, and $q_2 \neq f_X$.

[0058] Fig. 6 shows an example of the replacement operation. In Fig. 6, S_0 represents a start symbol, S represents a sentence, P represents a postposition, PP represents a postpositional phrase, NP represents a noun phrase, V represents a verb, VP represents a verb phrase, and $ represents a full stop. The left side of Fig. 6 shows a replacement operation where an arc whose input label is PP is replaced by a network Mpp representing a certain set of grammar rules having PP in the left side hand, and the right side of Fig. 6 shows corresponding parse trees.

[0059] On the whole, the replacement operation can be continued endlessly. However, memory in a computer implementing the finite state transducer generator is limited, and the size of the finite state transducer which can be generated is also limited. In the embodiment, a threshold value is set regarding the number of arcs representing the size of the finite state transducer.

When the number of arcs reaches a threshold value λ (in other words, when the finite state transducer reaches a specified size by repeating the arc replacement operation), the arc replacement operation is terminated, thereby realizing the finite state transducer with approximately.

[0060] (Process of determining an arc replacement order utilizing statistical information in the priority calculating part 4)

[0061] Through the arc replacement operation performed in the arc replacement part 3, the finite state transducer for use in incremental parsing can be generated. However, simply repeating the replacement operation alone may cause a problem that may terminate the replacement operation before a necessary arc is replaced. Therefore, when the replacement operation is performed, selection of arcs to be replaced is crucial. Using the statistical information as to frequency of applying grammar rules stored in the statistical information memory storage 11, the priority calculating part 4 determines an arc replacement order from relationship between arcs in the finite state transducer and nodes of a parse tree, based on that an arc corresponding to a node with higher derivation probability needs to be replaced.

[0062] The relationship between arcs in the finite state transducer and nodes of a parse tree will be described. The arcs in the finite state transducer are generated by recursively performing a network-based replacement operation starting from an arc whose input label is S_0. As a network represents a set of grammar rules, it can be considered that the grammar rules are applied to the arcs. On the other hand, when a parse tree is generated with a top-down procedure in the context-free grammar, the nodes are generated by applying the grammar rules first to S_0 to generate a node and recursively applying the grammar rules to the generated node. That is, both arcs and nodes are generated by recursively applying the grammar rules starting from the start symbol. The grammar rule application operation to the arc can be associated with that to the nodes. Thus, the arcs and nodes generated through the operation can be associated with each other. Fig. 6 shows an example of an arc-to-node correspondence using numbers. For example, an arc and a node indicated by number 1 in the figure are generated by applying the grammar rules in the following order: $S_0 \rightarrow S\$$, $S \rightarrow \ldots VP$, $VP \rightarrow PP\ V$. Thus, the arcs and nodes are associated with each other.

[0063] To generate a parse tree including a certain node in the parsing utilizing the finite state transducer, an arc corresponding to the certain node should be replaced. As the number of arcs to be generated is limited, however, not all of arcs are finally replaced. That is, not every parse tree can be generated. To generate a finite state transducer that can generate parse trees as much as possible, the arc replacement order should be considered. An index to determine the arc replacement order is referred to as a replacement priority. A parse tree including a node with a high derivation probability is more frequently generated. Therefore, it is considered that an arc corresponding to such a node should be replaced in preference to other arcs. A replacement priority value is set to a derivation probability of a corresponding node. When the finite state transducer is generated, the replacement priority is calculated for each of all arcs whose input labels are non-terminal symbols, using the statistical information regarding the frequency of applying the grammar rules stored in the statistical information memory storage 11, and the arc replacement operation is applied to the arcs in descending order of the arc replacement priority value in the arc replacement part 3.

[0064] Next, the calculation to obtain the derivation probability of a node will be described. Nodes of a parse tree are generated by applying the grammar rules to each node on a path from the root node S_0 to the node in order. The derivation probability is defined as a probability that the grammar rules are applied to each node in order on a path from S_0 to a node whose derivation probability is desired. In Fig. 7, node $X_{rM(1M)}$ is generated as follows: grammar rule r_1 is applied to the root node S_0 of the parse tree to generate nodes, grammar rule r_2 is applied to node $Xr_{1(11)}$ that is the 1_1-th node from the left of the nodes generated by the grammar rule r_1, and finally grammar rule r_M is applied to a node that is the 1_{M-1}-th node from the left of the nodes generated by grammar rule r_{M-1}. The derivation probability P ($Xr_{M(1M)}$) for the node $Xr_{M(1M)}$ is determined by:

$$\begin{aligned} P\left(X_{r_{M(l_M)}}\right) &= P\left(r_{1(l_1)}, r_{2(l_2)}, \ldots, r_{M-1(l_{M-1})}, r_{M(l_M)}\right) \\ &= P\left(r_{1(l_1)}\right) \\ &\times P\left(r_{2(l_2)} \mid r_{1(l_1)}\right) \\ &\times P\left(r_{3(l_3)} \mid r_{1(l_1)}, r_{2(l_2)}\right) \\ &\vdots \\ &\times P\left(r_{M(l_M)} \mid r_{1(l_1)}, \ldots, r_{M-1(l_{M-1})}\right) \end{aligned}$$

wherein $r_i(l_i)$ represents that grammar rule r_i is applied and grammar rule r_i+1 to be applied next is applied to a node generated by the (l_i)-th element of the right side of r_i. The reason why the position where the grammar rule is applied needs to be considered is because the grammar rules to be applied are different according to positions even in the same category. For example, when grammar rule N→NN is used, applicable grammar rules are different between the first N and the second N of the right hand side.

[0065] In the above expression, the value for $P(r_{i(1i)}|r_{1(11)}, \ldots, r_{i-1(1i-1)})$ is not affected by the applied position of the following grammar rule. Thus, the above expression can be rewritten by:

$$\begin{aligned} P(X_{r_{M(l_M)}}) &= P(r_{1(l_1)}, r_{2(l_2)} \ldots, r_{M-1(l_{M-1})}, r_M) \\ &= P(r_1) \\ &\times P(r_2 \mid r_{1(l_1)}) \\ &\times P(r_3 \mid r_{1(l_1)}, r_{2(l_2)}) \\ &\vdots \\ &\times P(r_M \mid r_{1(l_1)}, \ldots, r_{M-1(l_{M-1})}) \end{aligned}$$

[0066] The probability to derive a node is found in this way. However, if a grammar rule application probability is found from all grammar rules applied to find the derivation probability of a node as in expression 8, a data sparseness problem may arise, so that a finite state transducer to be generated is apt to depend on learning data. In the priority calculating part 4, the probability which the grammar rules are applied to a certain node depends on the grammar rules which have been applied to N-1 nodes tracing back in order from the certain node and the positions where the grammar rules have been applied. The obtained application probability is smoothed using a low-level conditional probability and liner interpolation.

[0067] A method for calculating the approximate probability P of applying the grammar rules will be described. The approximate probability P is determined by:

$$P\left(r_i \mid r_{i-N+1(l_{i-N+1})}, \ldots, r_{i-1(l_{i-1})}\right)$$

[0068] When grammar rules are applied to a certain node, nodes on a path from the root node S_0 to the certain node are traced in order, so that a N-1-tuple that pairs an applied grammar rule with a position on the right side of the applied grammar rule where a subsequent grammar rule is applied, is obtained. The currently applied grammar rule is matched with the N-1-tuple, and the certain node can be represented by a N-tuple ($r_{1(1i)}$, …, $r_{N-1(1N-1)}$, r_N). In Fig. 8, for example, a parse tree is generated by applying six grammar rules. Six groups are obtained from the parse tree. When N = 3, six 3-tuples are obtained as shown in Fig. 8. It is assumed that a null rule (#) is applied to nodes located above the start symbol of the parse tree.

[0069] Using a set of N-tuples obtained from learning data, the probability that grammar rule r_N with a condition of ($r_{1(1i)}$, …, $r_{N-1(1N-1)}$) is applied is determined by:

$$P(r_N \mid r_{1(l_1)}, \ldots r_{N-1(l_{N-1})}) = \frac{C(r_{1(l_1)}, \ldots r_{N-1(l_{N-1})}, r_N)}{\sum_{r_N} C(r_{1(l_1)}, \ldots r_{N-1(l_{N-1})}, r_N)}$$

wherein C(X) is the number of occurrences of X.

[0070] To obtain the probability of applying the grammar rules, linear interpolation values may be used. The linear interpolation values may be obtained by:

$$\begin{aligned}
&\hat{P}_N(r_N \mid r_{1(l_1)}, \ldots, r_{N-1(l_{N-1})}) \\
&= \lambda_N P_N(r_N \mid r_{1(l_1)}, \ldots, r_{N-1(l_{N-1})}) \\
&+ \lambda_{N-1} P_{N-1}(r_N \mid r_{2(l_2)}, \ldots, r_{N-1(l_{N-1})}) \\
&\vdots \\
&+ \lambda_2 P_2(r_N \mid r_{N-1(l_{N-1})}) \\
&+ \lambda_1 P_1(r_N \mid LHS(r_N))
\end{aligned}$$

wherein λ_1, …, λ_N are interpolation coefficients, and LHS(r_N) represents the left side category of r_N. Any condition except for P1(r_N|LHS(r_N)) does not include LHS(r_N). This is because it is clear that the category in the position 1_{N-1} of the grammar rule r_{N-1} is LHS(r_N).

[0071] Finally, in this procedure, the derivation probability for a certain node is determined by:

$$P(X_{r_{M(l_M)}}) = \prod_{i=1}^{M} \hat{P}(r_i \mid r_{i-N+1(l_{i-N+1})}, \ldots, r_{i-1(l_{i-1})})$$

[0072] In consequence of the integration of the states in the recursive transition network, the arcs generated from plural grammar rules exist in the recursive transition network. Therefore, one arc corresponds to two or more nodes of the parse tree in some case. In this case, the sum of the derivation probabilities of all the corresponding nodes is used as the derivation probability.

[0073] (Process of eliminating arcs labeled with non-terminal symbols in the arc eliminating part 4)

[0074] In the above process of generating the finite state transducer performed in the arc replacement part 3, when the number of arcs reaches threshold λ, the replacement operation is immediately terminated, and non-terminal symbols that were not replaced by the network remain unchanged in the finite state transducer. With the parsing of the embodiment, however, a state transition is made only when a part of speech of an input label of an arc matches a part of speech of a word inputted in the system, and any arcs whose input label is a non-terminal symbol are not used during parsing. Therefore, leaving these arcs is wasteful, and eliminating such arcs does not cause any problems. In fact, further performing the arc replacement while eliminating such arcs will improve the parsing capability of the finite state transducer. The following describes a process for eliminating arcs labeled with non-terminal symbols and further continuing performing the arc replacement operation.

[0075] First, the finite state transducer is generated by the process of the arc replacement part 3. When the number of arcs reaches threshold λ, the replacement operation is immediately terminated, and the following procedure is executed.

[0076] (Procedure to eliminate arcs whose input label is a non-terminal symbol)

[0077] Step A1: Arc e which has the highest replacement priority is selected from arcs labeled with non-terminal symbols as an arc to be replaced next. Input label of the arc e is I(e).

[0078] Step A2: It is checked whether replacement of arc e is valid. If it is not valid, arc e is eliminated and the procedure returns to step A1.

[0079] Step A3: Arcs in the finite state transducer, whose input labels are non-terminal symbols, are eliminated in order of ascending replacement priorities. The number of arcs to be eliminated is represented by $\lambda-((\text{the number of arcs in the finite state transducer}) - (\text{the number of arcs included in } M_{I(e)}) - 1)$. When the obtained number is negative, the arc is not eliminated.

[0080] Step A4: Arc e is replaced by network $M_{I(e)}$.

[0081] Step A5: If any arc whose input label is a non-terminal symbol remains in the finite state transducer, the procedure repeats steps A1 to A4.

[0082] In step A2 for validity check, arc e is checked as to whether there is an arc where the state at the start point of arc e is a transition destination, whether there is an arc where the state at the end point of the arc e is a transition source, whether the state is the initial state, or whether the state is the final state. If neither one is applicable, arc e is not analyzed, so that it is eliminated.

[0083] Through this operation, among the remaining arcs, arcs having higher replacement priority are further replaced, and arcs with lower replacement priority are eliminated. However, after the arcs are eliminated, any arcs cannot be reached from the initial state or cannot reach the final state will appear. These arcs cannot be used for parsing either. Therefore, when an arc is eliminated, the implications of the arc elimination are investigated. If an unusable arc further appears, the arc is eliminated together with arcs with lower replacement priority. When an arc is eliminated, the following is performed.

[0084] (A method to eliminate unnecessary arcs)

[0085] When an arc is eliminated, the following are checked as to every arc that shares the states of the start point and end point of the arc. If any one of the following conditions is applicable, the arc is eliminated according to the corresponding instruction. As to the eliminated arc, the same operations are recursively performed.

[0086] Step B1: When there is no arc that shares the start point of the eliminated arc as a transition destination, every arc that shares the start point of the eliminated arc as its start point is eliminated.

[0087] Step B2: When there is no other arc that shares the start point of the eliminated arc as a transition source, every arc that shares the start point of the eliminated arc as its end point is eliminated.

[0088] Step B3: When there is no other arc that shares the end point of the eliminated arc as a transition destination, every arc that shares the end point of the eliminated arc as its start point is eliminated.

[0089] Step B4: When there is no arc that shares the end point of the eliminated arc as a transition source, every arc that shares the end point of the eliminated arc as its end point is eliminated.

[0090] The above steps B1 to B4 are illustrated in Fig. 9. In Fig. 9, arcs indicated by a dotted line represent nonexistent arcs in each pattern. In each figure, as the arcs indicated by a dotted line are not existent when a central arc indicated by an "X" mark is eliminated, arcs further eliminated are also indicated by an "X" mark.

[0091] As a result of performing each process in the recursive transition network generating part 2, the arc replacement part 3, the priority calculating part 4, and the arc eliminating part 5, which are included in the finite state transducer generator 1, a finite transducer for use in incremental parsing is obtained.

[0092] (Incremental Generation of parse tree by an incremental parsing apparatus 21)

[0093] An incremental parsing apparatus 21 utilizing the finite state transducer 22 generated by the finite state transducer generator 1 will be described with reference to Fig. 10.

[0094] The incremental parsing apparatus 21 is made up of an input device 31, the finite state transducer 22, a connecting part 23, and an output device 32. The incremental parsing apparatus 21 is realized by a computer, which specifically includes CPU, ROM, RAM, a hard disk, a voice input device, and a display.

[0095] The input device 31 is designed to input a sentence to be parsed, and made up of a conventional sentence input device such as a voice input device or a keyboard. When sentences are inputted into the input device externally, the input device 31 inputs the sentences (word strings) into the finite state transducer 22 sequentially.

[0096] The finite state transducer 22 is a finite state transducer reflecting a result that a process of applying the grammar rules is previously calculated, and is generated by the above finite state transducer generator 1. The finite state transducer 22 makes a state transition for each word string inputted via the input device 31 and simultaneously outputs each piece of parse trees generated through the grammar rule application in order. The finite state transducer 22 is realized by that the CPU reads and executes the finite state transducer program stored in ROM or the hard disk.

[0097] The connecting part 23 sequentially connects each piece of the parse tree outputted by the finite state transducer 22. Thus, even in the middle of a sentence, the connecting part 23 can generate a parse tree for what has been inputted so far. The connecting part 23 is realized by that the CPU reads and executes a concatenation program stored in ROM or the hard disk.

[0098] The output device 32 outputs a parse tree generated by the finite state transducer 22 and the connecting part 23, as a result of parsing an inputted sentence. The output device 32 outputs a parsing result in the form of a file in RAM or the hard disk, or an indication on a display.

[0099] A process of generating parse trees incrementally in the incremental parsing apparatus 21 will be described. In the incremental parsing apparatus 21 of the embodiment, fundamentally words are successively inputted from the input device 31 to the finite state transducer 22, state transitions are made, and the parse trees are obtained. However, as the finite state transducer 22 generated by the finite state transducer generator 1 is non-deterministic, there is some possibility that two or more transition destinations exist as to an input. It is considered that in incremental parsing, a parsing structure should be outputted in accordance with each input. In the embodiment, a breadth first search is performed to generate a parse tree. The incremental parsing apparatus 12 has a list showing that the states and symbol strings each representing a parse tree outputted so far are linked in one on one relationship. When each word is inputted, all possible state transitions are made from the current state. At this time, the connecting part 23 connects a symbol string representing a parse tree for word string(s) inputted so far and an output label indicated with an arc in which a state transition is made, and a new parse tree is generated.

[0100] An example of actions in the incremental parsing apparatus 21 for Japanese will be described with reference to Fig. 11. A meaning in Japanese for each output symbol shown in Fig. 11 is put in parentheses as follows: S_0 (start symbol), S (sentence), NP(noun phrase), N-HUTU (common noun), HUTU-MEISI (common noun phrase), VAUX (verb phrase), VERB (verb), AUX (postpositional particle), AUX-DE (postpositional particle of Japanese, "de"), AUXSTEM (particle stem), AUXSTEM-MASU (particle stem of Japanese, "(gozai) masu"), INFL (conjugation ending), INFL-SPE-SU (conjugation ending of Japanese, "su"), and $(period).

[0101] Every time a word is inputted from the input device 31 to the finite state transducer 22, the finite state transducer 22 makes a state transition, and the output label of the arc where the state transition is made is connected by the connecting part 23. An output symbol string (which is a set of output labels connected) represents a parse tree. When a part of speech, for example, "HUTU-MEISI" (common noun) is inputted, its corresponding symbol string to be outputted represents a parse tree shown on the left side of Fig. 12. A parse tree shown on the right side of Fig. 12 represents a symbol string when input up to "AUX-DE" (postpositional particle of Japanese "de") has been done. In this way, the parse tree is expanded for every word input. In this example, a transition does not include ambiguity, and only one parse tree is outputted for each input of a particle of speech. However, as described above, when two or more transitions are possible, states and symbol strings are kept in pair, and parse trees corresponding to the number of transitions are made.

[0102] Next, an example of actions in another embodiment of the incremental parsing apparatus 21, which includes a finite state transducer generated with using the statistical information regarding frequency of applying English grammar rules, will be described with reference to Fig. 13. A meaning for each output symbol shown in Fig. 13 is put in parentheses as follows: S_0 (sentence), SQ(inversed yes/no question), VBZ(verb, 3rd person singular present), NP(noun phrase), DT(determiner), NN(noun, singular or mass), VP(verb phrase), VB(verb, base form), and $(period).

[0103] Every time a word is inputted from the input device 31 to the finite state transducer 22, the finite state transducer 22 makes a state transition, and the output label of the arc where the state transition is made is connected by the connecting part 23. An output symbol string (which is a set of output labels connected) represents a parse tree. When a part of speech, for example, "VBZ" is inputted, its corresponding symbol string to be outputted represents a parse tree shown on the left side of Fig. 14. A parse tree shown on the center of Fig. 14 represents a symbol string when input up to "DT" has been done. Further, a parse tree shown on the right side of Fig. 14 represents a symbol string when input up to "NN" has been done.

[0104] According to the finite state transducer generator 1, the arc replacement priority is calculated based on the statistical information regarding the frequency of applying the grammar rules, and the arc replacement operation is applied to arcs in descending order of the arc replacement priority, thus reliably generating a finite state transducer which can parse a great number of sentences within the limited size.

[0105] According to the embodiment, the finite state transducer generator 1 further includes the arc eliminating part 5. When the finite state transducer reaches a specified size, the arc replacement part 3 terminates the arc replacement operation.

Then, the arc eliminating part 5 eliminates arcs whose input labels are non-terminal symbols, which are not used for parsing, and simultaneously performs the arc replacement operation. This procedure also contributes to a generation of a finite state transducer which can parse a further great number of sentences.

[0106] According to the embodiment, the arc replacement operation is performed using a probability that grammar rules are applied to each node on a path from the start symbol to a certain node, as the arc replacement priority. Thus, the finite state transducer generator 1 can reliably a generator that can parse a considerable number of sentences.

[0107] In the incremental parsing apparatus 21, in other words, the finite state transducer 22 is generated by applying the arc replacement operation to the arcs starting from an arc having a highest priority obtained based on the statistical information regarding the frequency of applying the grammar rules. Using the finite state transducer of a limited size approximately transformed from the context-free grammar, the incremental parsing apparatus can parse a great number of sentences.

[0108] (Experiment)

[0109] Through the use of the finite state transducer generator 1 of the embodiment as describe above, a finite state transducer was generated, and the incremental parsing apparatus 21 was created by using the finite state transducer. To investigate the effect on incremental parsing in the incremental parsing apparatus 21, we conducted some experiments on parsing. We used a computer with the following specification for the experiments: Pentium® 4, 2 GHz of CPU and 2GB of memory. We used ATR speech database with parse trees (Japanese dialogue) as a learning data set and a test data set for the experiment. Using 9,081 sentences extracted at random from the speech database as the learning data set (statistical information as to frequency of applying grammar rules), we obtained the grammar rules and the application probability. At that time, there were 698 grammar rules, 337 particles of speech, and 153 categories. We used 1,874 sentences as the test data set. The average sentence length of the test data set was 9.4 words. We set the threshold value for the number of arcs of the finite state transducer to 15,000,000. This is because memory was used nearly to its maximum at the time of generation of the finite state transducer. The memory used during parsing was about 600 MB.

[0110] (Experimental results)

[0111] We conducted parsing on two parsing apparatuses to discuss comparisons of parsing time and parsing accuracy. One device was the incremental parsing apparatus 21 using the finite state transducer 22 of the embodiment (hereinafter referred to as a working example 1) and the other one was a parsing apparatus using a conventional incremental chart parsing (hereinafter referred to as a comparative example 1). The finite state transducer 22 of the working example 1 calculated a replacement priority and determined a replacement order using the grammar rule application probability when N = 3. N represents that a group of grammar rules used to find the probability is an N-tuple. The finite state transducer 22 of the working example 1 further eliminated arcs labeled with non-terminal symbols. As to the incremental chart parsing of the comparative example 1, a conditional probability was calculated and utilized for bottom-up parsing, based on the same principle as the grammar rule application probability used for generation of the finite state transducer. The product of the grammar rule application probabilities was calculated for each application of grammar rules. When the value exceeded 1E-12, applying of grammar rules was cancelled. Further applying of grammar rules was controlled with a possibility to reach an undecided term to be replaced. We set the parsing time for a word to 10 seconds on both the parsing apparatuses of the working example 1 and the comparative example 1. When the parsing time was over 10 seconds, parsing of the current word was terminated and parsing of the next word was processed. Table 2 shows parsing time and accuracy rate per word on both the parsing apparatuses of the working example 1 and the comparative example 1. The accuracy rate is a percentage of sentence including correct parse trees obtained as the parsing result from all sentences. A correct parse tree was a parse tree previously given to the sentence.

[0112]

Table 2: Experimental results of comparison between the working example 1 and the comparative example 1

	Parsing time (sec./word)	Accuracy rate (%)
Working example 1	0.05	87.5
Comparative example 1 (incremental chart parsing)	2.82	33.4

[0113] It is clear from the experimental results that the incremental parsing apparatus 21 of the working example 1 can process parsing faster than the comparative example 1. Where the average Japanese speech speed is about 0.25 seconds per word, the parsing speed of the working example 1 was 0.05 seconds per word, faster than the speech speed. This shows that the incremental parsing apparatus 21 of the working example 1 is effective in the real-time incremental parsing.

[0114] To make a comparison of the number of calculations for one word, we investigated parsing methods of the devices. In parsing according to the working table 1 using the finite state transducer, a calculation was counted each time a state transition was made to generate a parse tree. In the incremental chart parsing of the comparative example 1, a calculation was counted each time the grammar rules were applied, and a calculation was counted each time the tuple was replaced. As a result, the number of calculations for a word was 1,209 for the working example 1, and 36,300 for the comparative example 1, and thus, the number of calculations for the working example

1 was significantly lower than that for the comparative example 1. This experiment resulted in that it is possible to speed up the parsing process using the finite state transducer.

[0115] Next, we focused on an incremental parsing apparatus using a finite state transducer, and conducted experiments to investigate accuracy rates as a result of the parsing process. We prepared three examples of incremental parsing apparatuses. Working examples 2, 3 were incremental parsing apparatuses each including a finite state transducer generated with the replacement priority. A comparative example 2 was an incremental parsing apparatus including a finite state transducer generated without the replacement priority. The finite state transducer of the working example 2 was generated without elimination of arcs whose labels were non-terminal symbols. The finite state transducer of the working example 3 was generated with elimination of arcs whose labels were non-terminal symbols. As to the working examples 2, 3, each finite state transducer was generated by changing the number of conditions for the grammar rule application probability in the range from N=0 to N=4. N represents the number of conditions for the grammar rule application probability. The experiment results are shown in Fig. 13.

[0116] From the experiment results, we found that the accuracy rates of the working examples 2, 3 whose finite state transducers were generated with the replacement priority were greatly improved compared to the comparative example 2 whose finite state transducer was generated without the replacement priority, in other words, the control of the arc replacement order using the replacement priority was effective. The accuracy rate of the working example 3, whose finite state transducer was generated by eliminating the arcs labeled with non-terminal symbols, was improved, compared to the working example 2, whose finite state transducer was generated without arc removal. Therefore, the working examples 2, 3 showed improvements in accuracy as compared with the comparative example 2 and accuracy rate of nearly 90% was achieved with the combination of the replacement priority and removal of arcs labeled with non-terminal symbols. In addition, it is evident that the accuracy rate was improved as the number of conditions for the grammar rule application probability N was increased from 0 to 4.

[0106] While the invention has been described with reference to a specific embodiment, the description of the embodiment is illustrative only and is not to be construed as limiting the scope of the invention. Various other modifications and changes may occur to those skilled in the art without departing from the spirit and scope of the invention.

[0107] In the embodiment, the incremental parsing apparatus 21 is used alone, however, it may be installed in a simultaneous interpretation system or a voice recognition system, thereby realizing the simultaneous interpretation system or voice recognition system so as to work more simultaneously and precisely. When a voice recognition system including the incremental parsing apparatus 21 is installed in a robot, a rapid-response voice input robot or interactive robot can be realized. The incremental parsing apparatus 21 can be installed in automated teller machines (ATMs) placed in financial institutes, car navigation systems, ticket selling machines and other machines.

[0108] With the use of a context-free grammar written in a desired language (such as Japanese, English, and German) in the recursive transition network generating part 2, the finite state transducer 22 can be generated in accordance with the desired language. With the use of such a finite state transducer 22, the incremental parsing apparatus 21 can be structured in accordance with the desired language.